Molecular Surface Generation Using Marching Tetrahedra

SHEK LING CHAN,*,† ENRICO O. PURISIMA*

Biotechnology Research Institute, National Research Council of Canada, 6100 Royalmount Avenue, Montreal, Quebec H4P 2R2, Canada

Received 7 October 1997; accepted 12 March 1998

ABSTRACT: A method is presented to generate and triangulate molecular surfaces rapidly. It is based on the 'marching tetrahedra' approach. The method is fast, simple and easy to implement. Our approach is not analytical in nature. Hence no special treatment is required for complications with singularity, degeneracy, or with self-intersecting re-entrant surfaces. A quick test for determining the solvent accessibility of a point in space forms an important part of the method. This test has potential use outside of the surface generation algorithm such as in molecular field analysis where the solvent accessibility of a point needs to be determined. The triangulated surface generated is suitable for molecular graphics display as well as boundary element continuum dielectric calculations. J Comput Chem 19: 1268–1277, 1998

Keywords: molecular surface; marching tetrahedra; meshing

Introduction

For most practical purposes in molecular modeling, atoms are modeled as van der Waals spheres and molecules as interlocking atomic spheres. The external surface of these interlocking atomic spheres is the van der Waals surface of the molecule. For molecules of biological interest, solvation is always an important consideration. Therefore two more types of surfaces associated with a solvent probe sphere are commonly encountered in the literature. When a solvent probe sphere rolls around the molecule, the locus of its center defines the solvent accessible surface of the molecule as defined by Lee and Richards.[1] For the purpose of electrostatics considerations involving the solvent, it is also meaningful to define a surface which separates space that may be overrun by the solvent probe sphere from the part of space

*Correspondence may be addressed to either author.
†*Present address*: Tripos Inc., 1699 South Hanley Road, St. Louis, MO 63144.

CCC 0192-8651 / 98 / 111268-10

where solvent is excluded due to the presence of the molecule. Such a definition of a molecular surface was first put forward by Richards[2] and its mathematical description was subsequently thoroughly investigated by Connolly.[3–5] In this article, we shall follow Greer and Bush[6] to use the term solvent-excluded surface to refer to this definition of a molecular surface. A detailed description of the history about work on molecular surfaces can be found in a recent review by Connolly.[7]

As early as 1985, Connolly devised a method to generate and triangulate the solvent-excluded surface.[4] Complications arise from the need to deal with self-intersecting re-entrant surfaces. Zauhar and Morgan[8] developed a method to define and triangulate solvent-excluded surfaces that was later improved to provide a more robust triangulation for large molecules.[8,9] The self-intersecting re-entrant surface problem was dealt with by a redefinition of the surface at such places. GEPOL uses a different approach and builds approximations to solvent-excluded surfaces by repeatedly putting extra spheres in the space that is inaccessible to the solvent probe.[10–12] The surface can subsequently be triangulated based on a regular tessellation of the spheres.

In defining and triangulating solvent-excluded surfaces, self-intersecting surfaces, cusps and singularities pose special problems. Self-intersecting surfaces arise when two elements of concave re-entrant surfaces intersect each other back to back on two sides of the molecule. Cusps arise when two atoms are near enough so that a probe sphere cannot pass through the pair freely but they are far enough such that the neck-shaped surface between them breaks. In complex molecules such as proteins, it is not uncommon to have self-intersecting surfaces resulting in cusps and sharp ridges. Such cusps can cause difficulties for boundary element continuum dielectric calculations that utilize the triangulated surface. A detailed description of these cases and their remedies can be found in some references.[4,13]

There is another type of difficulty in dealing with solvent-excluded surfaces. A typical analytical solvent-excluded surface program calculates the vertices where a probe sphere can simultaneously touch three atoms and decide whether to have them registered by testing the accessibility of the vertex. The accessibility test is carried out by comparing the distance of the vertex to every atom with the radius of that atom. In singularity cases where a probe sphere can touch four or more atoms simultaneously, the distance from the vertex to the atom and the radius of the atom are equal. A wrong decision may be made concerning the accessibility of the vertex. This may lead to an inconsistent bookkeeping of the geometric elements and subsequently a wrong topology of the surface. Detailed descriptions of such situations can be found in some references.[13,14] Special treatments have to be employed to ensure topological consistency of the description.[15]

Here we are going to present a method that is capable of triangulating the solvent-excluded surface and the van der Waals surface of a molecule. It runs with competitive speed. The output can be used either for graphical display or for dielectric calculations using the boundary element approach.[8,16–22] The method by its nature avoids or minimizes many of the difficulties of dealing with singularity and re-entrant surface intersection and is hence simple and easy to implement.

Method

MARCHING TETRAHEDRA

The current method makes use of a marching tetrahedra approach.[23] Using marching tetrahedra, one can generate a contour surface for given a scalar function in 3D space. First, a tetrahedral tessellation of 3D space using a body-centered cubic lattice is generated as shown in Figure 1.

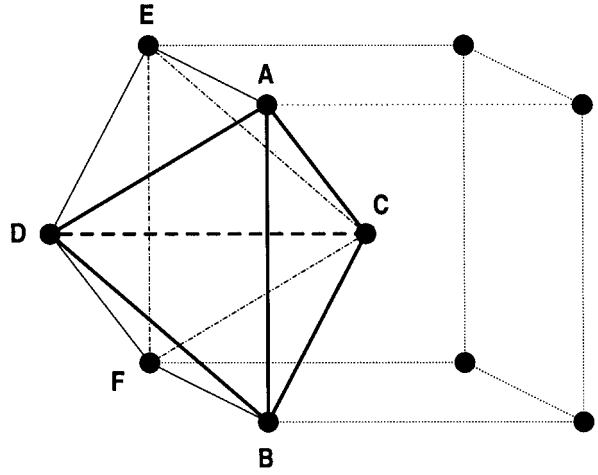

FIGURE 1. Tetrahedral element constructed from a body-centered cubic lattice. Points ABCD make up one tetrahedron. Point C is at the center of the cube shown and point D is at the center of the neighboring cube. Other tetrahedra shown are ACDE, CDEF, and BCDF.

Functional values are then sampled at the grid points. A contour surface is generated by examining the function values at the corners of each tetrahedron. If some vertices of a tetrahedron have functional values above the contour level while some have values below, an element of the contour surface is constructed across this tetrahedron. The intersection points of this surface element on the edges of the tetrahedron are obtained by interpolation. Figure 2 shows the two possibilities that can arise for generating a contour surface element.

The marching tetrahedra method is analogous to the marching cubes algorithm.[24,25] However, it has advantages over marching cubes. First of all, the use of cubic elements can result in an ambiguity as to the orientation of the surface elements cutting across the cube.[26,27] No such ambiguity exists with tetrahedral units. Secondly, there are many more types of intersections of the contour surface with the cube. Even after reduction by taking into account symmetries, there are 14 types of intersections some of which are rather complex.[24,25] This can be reduced to 5 types by using hexagonal prisms instead of cubes.[28] In contrast, there are only two kinds of intersections with tetrahedra (Fig. 2).

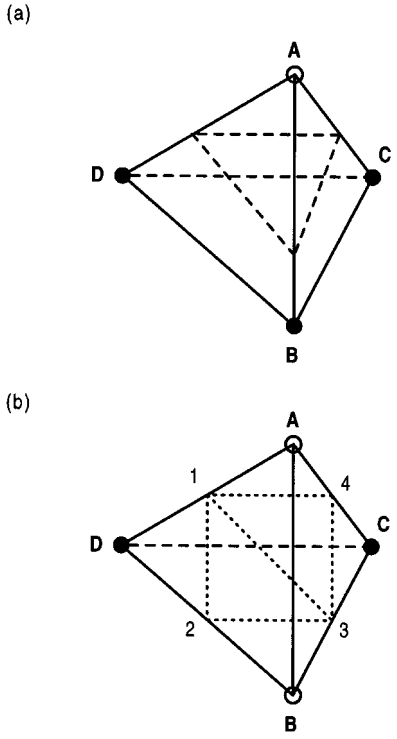

FIGURE 2. Two types of surface elements that can cut a tetrahedron. Filled and open circles indicate vertices inside and outside the solvent-excluded surface, respectively. (a) When exactly three vertices lie inside (or outside) the molecular surface, the surface cuts three edges of the tetrahedron and defines a triangular element as shown. (b) When exactly two vertices lie inside the molecular surface, the surface cuts four edges of the tetrahedron and defines two triangular elements, triangles 123 and 134.

There are other ways of tessellating 3D space with tetrahedra.[29,30] However, using the body-centered cubic lattice approach requires only one kind of tetrahedral unit and the tetrahedral unit generated has a good aspect ratio, i.e., the edge lengths are of similar magnitude.

MOLECULAR SURFACE

To construct a molecular surface, one could in principle use the electron wave function density to assign function values at the grid points and then subsequently construct the contour surface at an appropriate level. However, such an approach would be impractical for medium or large molecules. The conventional way is to use spheres to represent atoms. In such a representation, a grid point is then assigned as being inside or outside the molecular volume. The surface elements are then generated by an iterative interpolation to find the intersection with the tetrahedron edges.

As an example of the iterative step, let us examine the case of generating a van der Waals surface from the marching tetrahedra. Consider a tetrahedral unit that is cut by the surface. If one endpoint (A) of an edge lies inside the molecule while the other endpoint (B) lies outside, the surface cuts through this edge. We then check if the midpoint (C) of the edge lies inside or outside. If it lies inside, the point of intersection of the surface with this edge will lie within the segment BC. Subsequently the midpoint of BC can be tested. Similarly if C lies outside, the midpoint of AC can be tested. If l is the length of the edge, after n iterations, the locality of the intersection point can be narrowed down to a segment of length $l/2^n$. In our implementation, we simply take the midpoint of this final segment to be the intersection point. Using this iterative bisection, the intersection positions of the surface elements with the tetrahedral units are obtained.

For van der Waals surfaces, testing whether a point is inside the surface is a straightforward exercise. In fact, even the calculation of the intersection point of the surface with a tetrahedron edge can be done analytically without recourse to iteration. However, the test for a solvent-excluded

surface is not as simple. In order to use the marching tetrahedra to generate a molecular surface, we require an efficient test to decide whether a point lies inside the solvent-excluded volume or not. A method for doing this is presented in the next section.

ACCESSIBILITY TEST

Let us define a point P to be solvent accessible if and only if it can be engulfed within a solvent probe sphere that does not intersect with any atoms in the molecule. (To avoid confusion, it should be pointed out that a solvent-accessible point as defined here does not mean a point lying on the solvent-accessible surface.) The complement of the set of solvent-accessible points is the volume enclosed by solvent-excluded surface.

First of all, note that points P that lie inside the van der Waals radii of atoms are obviously inaccessible. Secondly, points P that lie outside a distance of $r + r_p$ from any atom of radius r and probe radius r_p (i.e., points that lie outside the solvent accessible surface as defined by Lee and Richards[1] are clearly outside the molecular surface). We only need to test points in the remaining region lying between the van der Waals surface and the solvent-accessible surface, which will be referred to as the shell layer.

Consider a spherical surface centered at P of radius r_p, hereafter called the celestial sphere. If a probe sphere can be placed with its center located at some point on the surface of the celestial sphere without intersecting any atoms in the molecule, P will be solvent accessible (Fig. 3). Conversely, if at no point on the celestial sphere can one place a probe sphere, P is in general not solvent accessible. Note that certain small cavities large enough to contain a probe sphere but with no direct access to the bulk solvent region will be flagged as inaccessible by this method (Fig. 4). But then for such small patches of solvent accessible volume, placing explicit solvent molecules there could possibly give a more accurate representation than treating that small space as a continuous solvent medium.[31]

We now describe a series of tests to determine the feasibility of placing a probe sphere on some point on the celestial sphere of P. Each atom A of radius r lying at a distance d away from P can occlude part of the celestial sphere from probe spheres if

$$d < r + 2r_p \qquad (1)$$

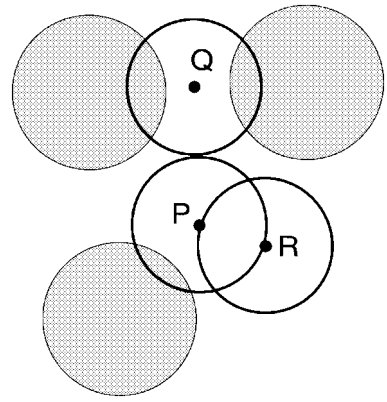

FIGURE 3. Celestial sphere of a point, P. P is solvent-accessible because there exists a point R on its celestial sphere where one can center a solvent probe that does not overlap any atom. In contrast, Q is not solvent-accessible because at no point on its celestial sphere can one center a solvent probe that does not overlap any atom.

The part of the celestial sphere that is occluded is the interior of a circular patch of angular radius θ given by

$$(r + r_p)^2 = d^2 + r_p^2 - 2dr_p \cos\theta \qquad (2)$$

A schematic illustration is given in Figure 5 where the cross-section of the occluding circle is highlighted. Atoms near point P give rise to a collection of occluding circular patches. The centers of

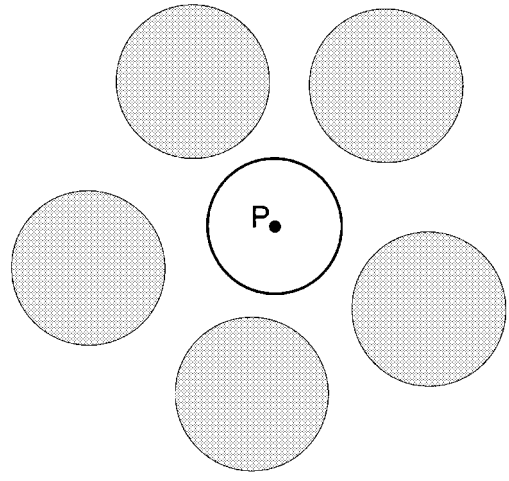

FIGURE 4. Small cavity missed by the algorithm. P lies outside the molecular surface since it is contained entirely in the solvent probe shown. However, our method flags it as solvent-inaccessible because at no point on its celestial sphere can one center a solvent probe that does not overlap any atom.

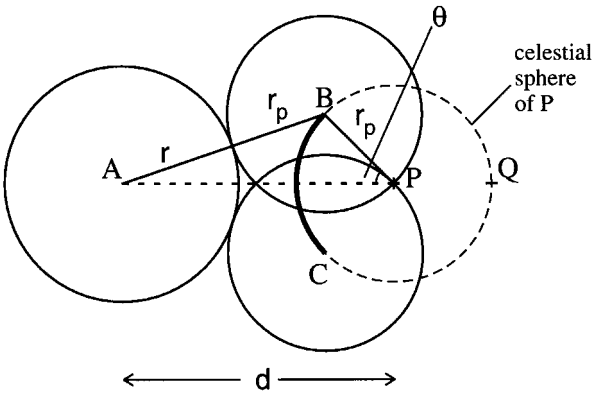

FIGURE 5. Occlusion of the celestial sphere of a point, *P*. *A* is the center of an atom with radius *r* located *d* Å away from *P*. r_p is the probe radius. BC is the region on the celestial sphere occluded by atom A. θ is the angular radius of the circle defining the boundary of the occluded region.

these occluding circles and their radii are calculated and stored for later testing. Meanwhile, during the calculation for each of these circles, we attempt to place a probe sphere on the celestial sphere at the opposite pole of the center of the circle (Point *Q* in figure 5). If this probe sphere does not intersect with any atoms, the point *P* is accessible and no further calculations need to be carried out. This heuristic is a quick way of identifying accessible points surrounding well-exposed convex surfaces of the molecule. This test will be referred to as criterion A.

Next, note that if the angular radii θ_1 and θ_2 of two occluding circles separated by an angular distance of ϕ (i.e., ϕ is the angle between the vectors pointing from *P* towards the centers of these two circles) are such that

$$\theta_1 + \theta_2 + \phi > 2\pi \quad (3)$$

then these two circles alone will occlude the whole celestial sphere. Again no further calculations need to be carried out to conclude that *P* is inaccessible. This will be referred to as criterion B. On the other hand, if

$$\theta_1 > \theta_2 + \phi \quad (4)$$

circle 2 will be enclosed within circle 1 and it does not need to be considered in subsequent tests.

During the calculations of the occluding circles, the connectivity of the circles (i.e., whether a pair of circles intersect) is also tabulated. After calculating and registering all occluding circles that need to be considered, if some circle does not intersect with any other circle, then points on the boundary of this circle will be feasible points and hence *P* is accessible. If some circle intersects only one other circle, then the intersection points between these two circles will be feasible and hence *P* is also accessible. These together will be referred to as criterion C.

If the accessibility of *P* still remains undetermined, then the intersection points of each pair of circles will be investigated. If *A* is an intersection point between circles 1 and 2, then each circle that intersects with both circle 1 and circle 2 is tested to see if it engulfs *A*. If *A* is not engulfed by any circles, then *A* is a feasible point on the celestial sphere and hence *P* is accessible. Finally, if all intersection points similar to *A* are engulfed, then the whole celestial sphere is occluded, and *P* is inaccessible (See Appendix A for a proof). This final test will be referred to as test D.

The algorithm has been implemented using the C++ computer language. Excluding the standard C libraries, the whole algorithm only takes less than 500 programming statements, as can be counted by the number of ';' in the program. To speed up calculations, lists of neighboring atoms are set up for each grid point. Whenever an iteration is carried out to determine the intersection point of the surface on an edge of the tetrahedral grid, only atoms on the lists of the two grid points at the two ends of this edge need be considered.

Results and Discussion

Efficacy of the Solvent-Accessibility Tests

We first examined the performance of each of the solvent-accessibility criteria in determining the solvation status of a grid of points defined by a body-centered cubic lattice. Crambin, BPTI and papain were used as test molecules. Crystal structures were obtained from the PDB. Hydrogen atoms were added using SYBYL (Tripos Inc.). Calculations were carried out for grid spacings of 0.4 to 1.4 Å (in 0.2 Å increments) on these molecules. In each case, the grid was dimensioned to be big enough to contain the whole molecule. Table I gives some statistics of the efficacy of criteria A, B, C and D in determining the solvent accessibility of points in the shell layer. For a given molecule, the results do not vary much for the different grid spacings. We see that over 90% of the points in the shell layer have been weeded out by criteria A and B. This is fortunate since these are the least expensive tests to carry out. Criterion C has a very weak

TABLE I.
Efficacy of Various Tests for Solvent Accessibility.

	Crambin	BPTI	Papain
Test A	79.9 −80.5%	77.5 −78.4%	66.4 −68.2%
Test B	13.7 −14.1%	14.8 −15.7%	22.8 −24.0%
Test C	0.2 −0.3%	0.2 −0.3%	0.2 −0.3%
Test D	5.5 −5.8%	6.5 −7.0%	8.8 −9.4%
n_{neigh}	22.4 −23.0	23.5 −23.8	26.2 −26.4
n_{circ}	7.7 −7.9	8.3 −8.5	9.6 −9.7

As discused in the text, tests A −D are applied successively to determine if a grid point is accessible or not. For each protein, the percentage of points in the shell layer whose accessibility is determined by the given test is shown. The range of percentages reported is for surfaces with grid spacings ranging from 0.6 Å to 1.4 Å in 0.2-Å increments. Also shown are the average number of atomic neighbors whose occluding circles theoretically need to be considered for test D. n_{circ} is the actual average number of occluding circles examined in test D. It is smaller than n_{neigh} due to large occluding circles completely engulfing smaller ones.

weeding power. But, since the test is part of the necessary checks before entering test D, its use does not lead to extra calculations.

Also shown in Table I are the average number of atomic neighbors of these points, i.e., atoms that are within a distance $2r_p + r$ away, where r_p is the probe radius and r is the atom's radius. There are an average of 23 to 26 of these atomic neighbors capable of occluding part of the celestial sphere of a given grid point. In principle, for points that require test D to determine their accessibility, one would have to examine the intersections of all pairs of the 23 to 26 corresponding circles on the celestial sphere. This would be a heavy computational cost. Fortunately, for points P in the shell layer region, there is at least one atom of radius r within a distance $r + r_p$ away. This atom will create a big occluding circle that covers roughly half, or more, of the celestial sphere. Many occluding circles due to atoms on this side of the point P will be engulfed by this circle and hence discarded according to equation (4). In practice, removal of these redundant circles results in an average of 8 or 9 circles that remain to be considered for points that require test D. This corresponds to about 35% of the original number of circles for these points. Considering that the number of tests in test D that needs to be carried out for each sphere is proportional to the square of the number of circles, this reduction in the number of circles contributes to the speed of the algorithm.

In the current algorithm, the time taken to test a point varies greatly from point to point. Points in the shell layer above the convex part of an atom are fast to deal with, because of criterion A. Points that lie very deep inside the crevices of the van der Waals atomic surface also do not cause much trouble, because of criterion B. It is points that are in between that takes up most of the time. For a regular three dimensional grid set up for practical computations, such points make up only a small percentage of all the points as seen above.

Molecular Surfaces

In this section we examine in more detail the nature of the molecular surfaces generated by the marching tetrahedra. We shall calculate the surface area and the volume of our triangulated surface and see how they vary as a function of the grid spacing and the number of iterations.

We first examined the simple case of two atoms of radii 1.8 Å separated by a distance of 4 Å. The van der Waals surfaces of these atoms do not touch each other but they are near enough such that a water probe sphere of radius 1.4 Å cannot pass through the strait between them resulting in a saddle-shaped solvent-excluded surface between them. The area of the system can be calculated analytically.

Figure 6 shows the area of the system as a function of the grid spacing and of the number of iterations. The horizontal line in the figure indicates the exact analytical value of the area. We note that for larger grid spacing the area is underestimated. This is due to the use of flat surface elements which underestimate the area of the curved surface elements that they approximate. As the grid spacing decreases, the error is reduced and the area approaches the theoretical value. Having too few iterations in the bisection interpolation procedure results in a more ''corrugated'' surface with some points lying significantly outside the molecular surface and others inside. This is the reason for the larger calculated areas for the low iteration calculations. Five iterations appears to be sufficiently converged. We note that with 5 iterations, a 0.6 Å grid spacing gives an area within 0.6% of the true value. At a grid spacing of 1.2 Å, we are still within 2% of the exact value.

Figure 6 also shows the volume of the diatomic system as a function of the grid spacing and number of iterations. We note that the calculated volume is less sensitive than the area with respect to the number of iterations. This is because for the corrugated surface generated there is a partial cancellation of errors. The volume is increasingly un-

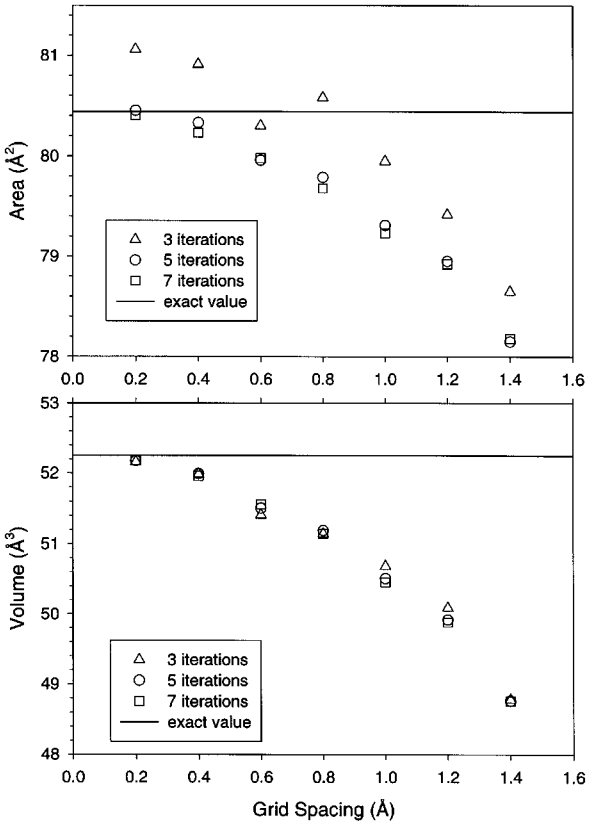

FIGURE 6. Dependence of area and volume on number of iterations and grid spacing: diatomic test case. Calculated areas and volumes for two atoms of radii 1.8 Å separated by a distance of 4 Å are shown.

derestimated as the grid spacing increases. This is due to the flat triangular surface elements used to approximate curved surfaces. For this simple system, the bulk of the surface is the convex van der Waals surface of the two atoms. Hence, the generated surface will tend to be mostly inscribed triangles and will underestimate the molecular volume. At grid spacings of 0.6 and 1.2 Å, the volume is underestimated by 1.4 and 4.5%, respectively.

The sensitivity of the calculations to the relative orientation of the molecule with respect to the grid was also investigated. There is little dependence of the result on the positioning of the grid as seen in Figure 7. There is negligible variation in computed area among different orientations for grid spacings of less than or equal to 1 Å.

We next describe the performance of the method on three test proteins: crambin, BPTI and papain. Random perturbations of the orientation of the molecule relative to the grid did not affect the results significantly (data not shown).

Figure 8 shows the calculated areas and volumes for crambin as a function of grid spacing and

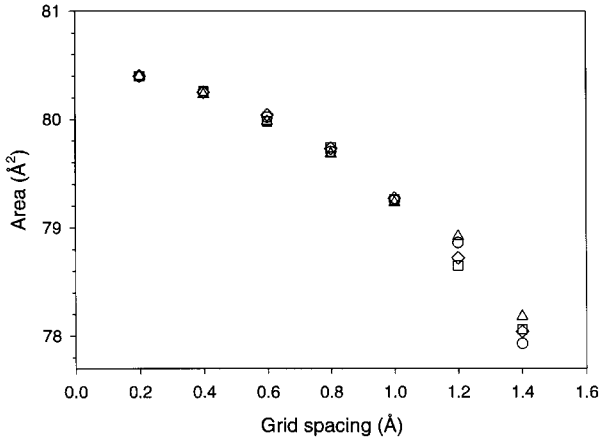

FIGURE 7. Dependence of area on the relative orientation of the grid: diatomic test case. At each grid spacing value, calculated areas for four different orientations of the diatomic molecule are shown.

number of bisection iterations. Calculations using 3, 5 and 7 bisection iterations are presented. We note that the trend with respect to the grid spacing and number of iterations parallels that of the two-atom example. Five iterations appears to be suffi-

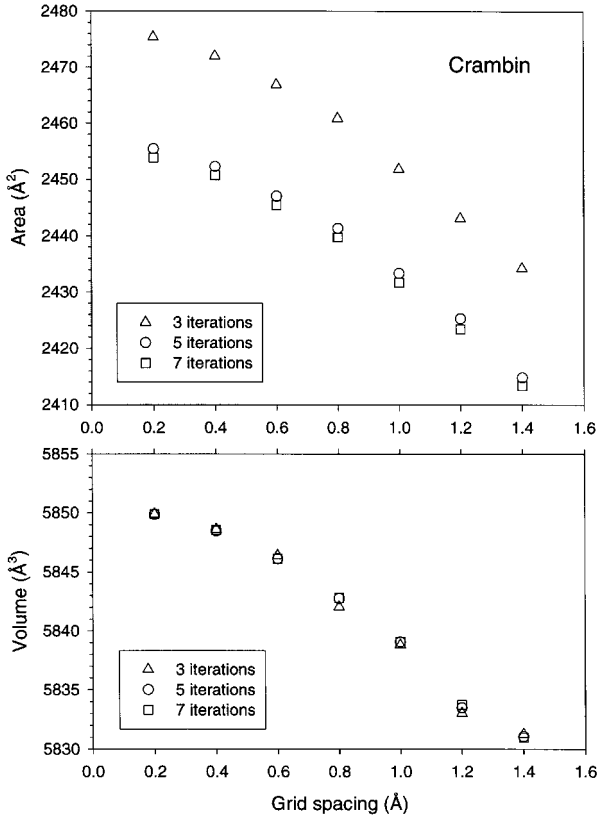

FIGURE 8. Dependence of area and volume on number of iterations and grid spacing: crambin test case.

cient in placing a mesh point on the surface. We see that for the range of 0.2 to 1.4 Å grid spacing, the variation in calculated areas is of the order of 1 to 2%. For the calculated volumes, the variation is even smaller at about 0.3%. As would be expected, CPU times increase with the number of iterations and with reduced grid spacing size (Figure 9). The use of 5 iterations and a grid spacing of 1.0 to 1.4 Å appears to be a good compromise between speed and accuracy. Figures 10 and 11 show the results for BPTI and papain, respectively. Again, even at coarser spacings of 1.0 to 1.4 Å, the difference in calculated areas as compared to finer grid spacings appears to be less than 2%. The difference in calculated volumes between 0.4 and 1.4 Å grid spacings are 0.3 and 0.07% for BPTI and papain, respectively. For large systems such as proteins, the error in the volume is small since most of the volume is made up of highly buried atoms.

To give a visual impression of the generated surfaces, figures 12a and 12b give the solvent-excluded surface of BPTI generated using grid spacings of 0.8 Å and 0.5 Å, respectively.

Alternate Solvent-Accessibility Test

An alternate method to determine solvent accessibility of points in the shell layer has been reported by You and Bashford.[32] Their method first defines all geometric entities involved in describing the solvent accessible volume, and then tests whether a point lies inside or outside these entities. A substantial amount of computer time is spent to define the geometric entities. However, subsequent accessibility tests for the grid points are fast once these entities are defined.

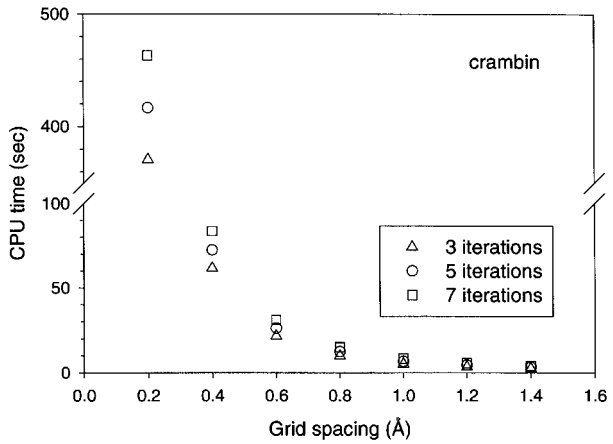

FIGURE 9. Dependence of CPU time on number of iterations and grid spacing: crambin test case. CPU times are for a Silicon Graphics workstation with a 195 MHz MIPS R10000 processor.

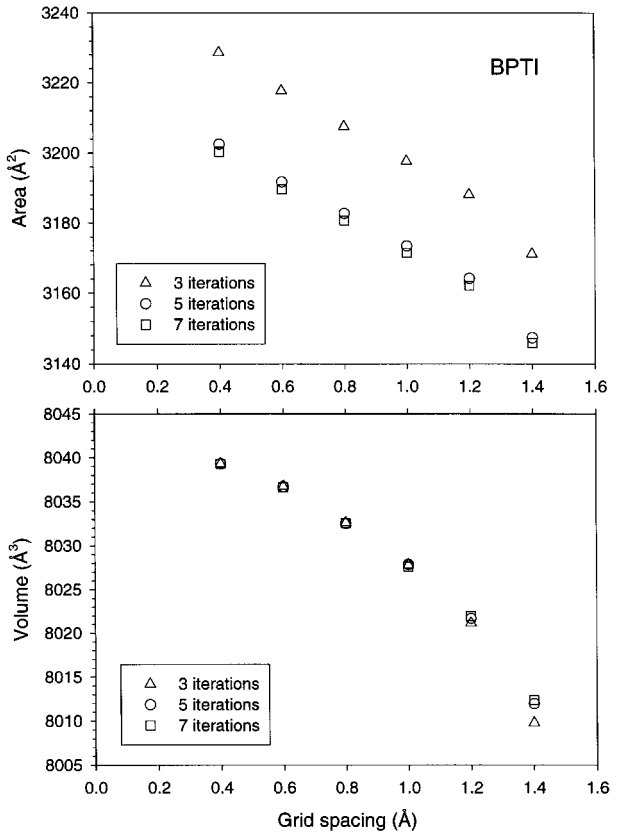

FIGURE 10. Dependence of area and volume on number of iterations and grid spacing: BPTI test case.

We have tried to incorporate the accessibility test of You and Bashford[32] in the marching tetrahedra program. (A copy of their program is available through the gnu public domain.) In our hands, we find that grafting their routine for testing the accessibility of a single point into our program gave rather slow execution times. This appears to be due to some repetitive calculations in these routines unnecessary for our purposes. We have rewritten this routine to make it more efficient for our application and incorporated it into our algorithm as a parallel to the program with our own accessibility test. Aside from helping us to validate our program by comparing the final outputs, it also allows us to compare the performance of the two methods.

Table II gives the speed for computing the accessibility of cubic grids for these molecules using our method and using the method of You and Bashford. The first point to note is that with the YB method, the CPU time is essentially independent of the grid spacing since the bulk of the CPU time

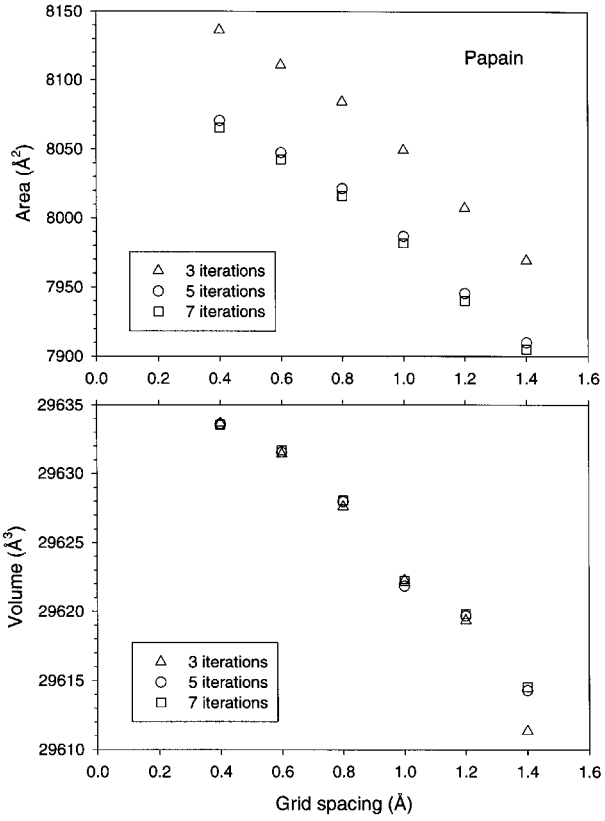

FIGURE 11. Dependence of area and volume on number of iterations and grid spacing: Papain test case.

TABLE II.
Comparison of Solvent-Accessibility Tests.

Grid spacing (Å)	CPU (sec)[a]	CPU (sec)[b]
BPTI		
1.0	7.4	1.2
0.8	8.2	2.7
0.5	8.8	10.4
Papain		
1.0	42	5
0.8	43	9
0.5	44	23

The CPU times reported in this table are for an SGI workstation with a 200-MHz Mips R4400 processor.
[a]Method of You and Bashford.[32] [b]Current method.

is spent on the calculation of the geometric quantities that depend only on atomic coordinates. In their method, once the geometric entities are defined, accessibility tests are quick. For our method, the time required is roughly proportional to the number of points involved, hence, the dependence on grid spacing. Overall, it seems that when a single calculation for a given grid is required, our method is faster, especially for coarser grid spacings. However, when multiple calculations are required, such as that involved in focusing in finite-difference continuum dielectric calculations, our

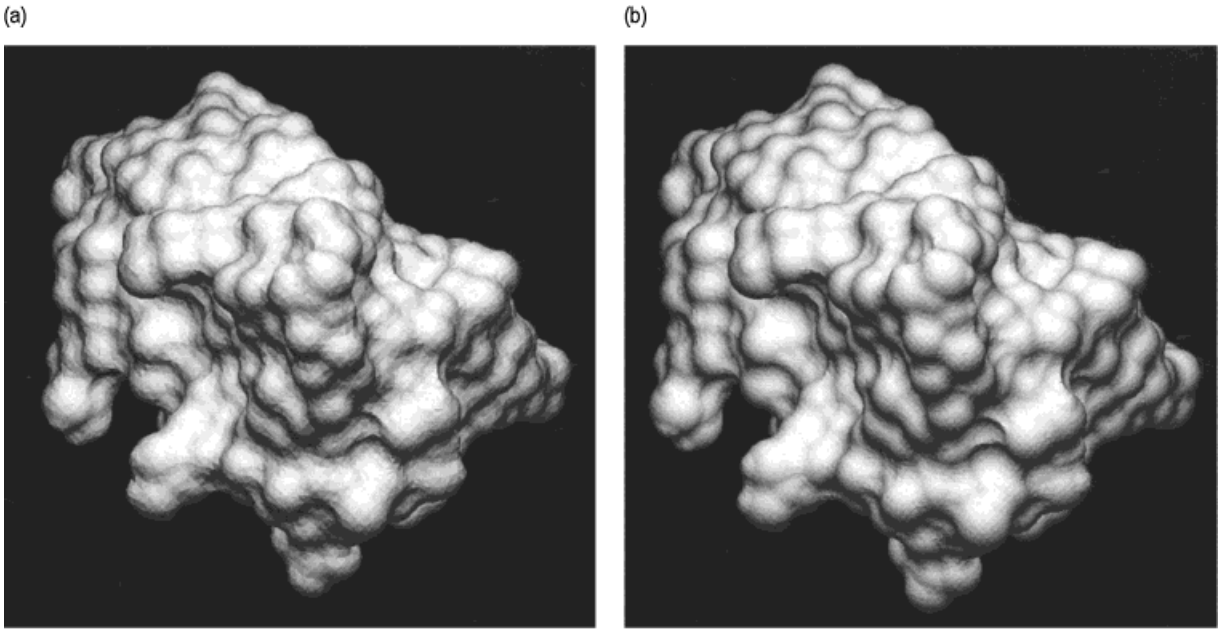

FIGURE 12. Molecular surface generated for BPTI using grid spacings of (a) 0.8 Å, and (b) 0.5 Å.

method is less competitive than the method of You and Bashford.

Conclusion

A method to triangulate solvent-excluded surfaces has been presented. The method is based on the ''marching tetrahedra'' algorithm. The method avoids complications of degeneracy, cusps and self-intersecting surfaces present in most analytical treatments of the solvent-excluded surface. Applications of the method include graphical display of molecular surfaces, molecular surface area calculation, and boundary-element continuum dielectric calculations.

Acknowledgments

The authors would like to thank Tony You and Donald Bashford for making their solvent-accessibility program available. We thank one of the referees for suggesting the addition of Figures 3 and 4 to clarify the discussion. This is publication number 41424 of the National Research Council of Canada.

Appendix A. Proof of Accessibility Test D

If there are inaccessible circles, there is at least one inaccessible patch on the celestial sphere. Suppose there is a part of the celestial sphere that is accessible. Then the boundary of the inaccessible patch will either be a circle or be made up of arcs of circles. If the boundary is a circle, then this circle will not be connected to any other circles. If the boundary is made up of a number of arcs, then some intersection points between certain arc pairs must be accessible. Hence if all circles are connected to at least two circles and there is no intersection point accessible, then there cannot be any part of the celestial sphere that is accessible.

References

1. B. Lee and F. M. Richards, *J. Mol. Biol.*, **55**, 379 (1971).
2. F. M. Richards, *Ann. Rev. Biophys. Bioeng.*, **6**, 151 (1977).
3. M. L. Connolly, *J. Appl. Cryst.*, **16**, 548 (1983).
4. M. L. Connolly, *J. Appl. Cryst.*, **18**, 499 (1985).
5. M. L. Connolly, *Science*, **221**, 709 (1983).
6. J. Greer and B. L. Bush, *Proc. Natl. Acad. Sci. USA*, **75**, 303 (1978).
7. M. L. Connolly, *Network Science*, **2** (1996). http://www.awod.com/netsci/Science/Compchem/feature14.html
8. R. J. Zauhar and R. S. Morgan, *J. Comp. Chem.*, **11**, 603 (1990).
9. R. J. Zauhar, *J. Comput.-Aided Mol. Design*, **9**, 149 (1995).
10. J. L. Pascual-Ahuir and E. Silla, *J. Comp. Chem.*, **11**, 1047 (1990).
11. E. Silla, I. Tuñón, and J. L. Pascual-Ahuir, *J. Comp. Chem.*, **12**, 1077 (1991).
12. J. L. Pascual-Ahuir, E. Silla, and I. Tuñón, *J. Comp. Chem.*, **15**, 1127 (1994).
13. M. F. Sanner, A. J. Olson, and J.-C. Spehner, *Biopolymers*, **38**, 305 (1996).
14. V. Gogonea and E. Osawa, *Supramolecular Chemistry*, **3**, 303 (1994).
15. F. Eisenhaber and P. Argos, *J. Comp. Chem.*, **14**, 1272 (1993).
16. S. Miertus, E. Scrocco, and J. Tomasi, *Chem. Phys.*, **55**, 117 (1981).
17. A. A. Rashin and K. Namboodiri, *J. Phys. Chem.*, **91**, 6003 (1987).
18. R. J. Zauhar and R. S. Morgan, *J. Comp. Chem.*, **9**, 171 (1988).
19. R. J. Zauhar and R. S. Morgan, *J. Mol. Biol.*, **186**, 815 (1985).
20. R. J. Zauhar and A. Varnek, *J. Comp. Chem.*, 17, 864 (1996).
21. B. J. Yoon and A. M. Lenhoff, *J. Comp. Chem.*, **11**, 1080 (1991).
22. E. O. Purisima and S. H. Nilar, *J. Comp. Chem.*, **16**, 681 (1995).
23. S. L. Chan and E. O. Purisima, *Computers and Graphics*, **22**, 83 (1998).
24. W. E. Lorensen and H. E. Cline, *Computer Graphics*, **21**, 163 (1987).
25. W. Heiden, T. Goetze, and J. Brickmann, *J. Comp. Chem.*, **14**, 246 (1993).
26. M. J. Düurst, *Computer Graphics*, **22**, 72 (1988).
27. J. Wilhelms and A. Van Gelder, *Computer Graphics*, **24**, 79 (1990).
28. X. Zhexin, S. Yunyu, and X. Yingwu, *J. Comp. Chem.*, **16**, 512 (1995).
29. Y. Zhou, W. Chen, and Z. Tang, *Computers and Graphics*, **19**, 355 (1995).
30. A. Guéziec and R. Hummel, *IEEE Transactions on Visualization and Computer Graphics*, **1**, 328 (1995).
31. S. L. Chan and C. Lim, *J. Phys. Chem.*, **98**, 692 (1994).
32. T. You and D. Bashford, *J. Comp. Chem.*, **16**, 743 (1995).